

Dynamic Simulation of Autonomous Legged Locomotion

Michael McKenna and David Zeltzer

Computer Graphics and Animation Group
The Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA

ABSTRACT

Accurate simulation of Newtonian mechanics is essential for simulating realistic motion of jointed figures. Dynamic simulation requires, however, a large amount of computation when compared to kinematic methods, and the control of dynamic figures can be quite complex. We have implemented an efficient forward dynamic simulation algorithm for articulated figures which has a computational complexity linear in the number of joints. In addition, we present a strategy for the coordination of the locomotion of a six-legged figure – a simulated insect – which has two main components: a gait controller which sequences stepping, and motor programs which control motions of the figure by the application of forces. The simulation is capable of generating gait patterns and walking phenomena observed in nature, and our simulated insect can negotiate planar and uneven terrain in a realistic manner. The motor program techniques should be generally applicable to other control problems.

1. COORDINATING AND CONTROLLING JOINTED FIGURE MOTION

Realistic modeling and animation of human and animal figures has long been a goal of researchers in computer graphics. There are two aspects to the synthesis of motor behavior [1]:

- The **Coordination** problem. How do we organize the movements of a complicated figure into coherent, useful motions? In other words, given a jointed figure with many degrees of freedom (DOFs), how do we decide which DOFs are needed to perform a given motion, and how do these movements vary with respect to each other?

- The **Control** problem. Given a set of DOFs, find appropriate time-varying parameters for each DOF necessary to effect a given motion.

In this paper we will describe approaches to both problems. We use *physically-based* methods with dynamic motor control

This work was supported in part by National Science Foundation Grant IRI-8712772, and equipment grants from Hewlett-Packard Co., Gould Electronics, Inc., and Apple Computer, Inc.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

to generate realistic movements, and we use *biologically-based* mechanisms to coordinate complex patterns of movement. In particular, we have implemented a dynamics simulation algorithm based on the work of Featherstone [2, 3], which is more efficient than dynamic simulation algorithms previously reported in the computer graphics literature. We use this algorithm to compute the movements of a six-legged virtual insect. Based on results from ethology and physiology, we coordinate its adaptive gait over planar and uneven terrain using coupled-oscillators, reflexes and a set of motor programs which apply controlling forces.

In the next section, we discuss related work. In Section 3 we describe the major components of our dynamic locomotion simulator. We conclude with a discussion of results we have obtained so far.

2. RELATED WORK

2.1. Dynamics, Forward and Key-Event Simulation

Rigid-body simulators compute the motion of articulated figures comprised of rigid links connected by joints, which can move relative to each other with one or more DOFs. Constraint-based approaches define the relationships among the links, and then solve for the forces required to maintain the prescribed connections. These methods are typically computationally more expensive and numerically less stable than non-constraint-based techniques [4, 5].

Isaacs and Cohen describe a straightforward method of constraint simulation based on a matrix formulation [6]. Joints are configured as kinematic constraints, and either accelerations or forces can be specified for the links. An equation is established for each DOF, yielding n simultaneous equations to solve, giving $O(n^3)$ complexity. Interdependencies among the constraints typically make the matrix non-sparse, so that sparse matrix solutions cannot be employed to reduce the complexity. Control is applied through the kinematic constraints and user-defined "behavior functions," which specify accelerations or forces over time. Arbitrary motions can be specified, however, which have no physical basis.

Barzel and Barr describe constraint-based simulators which allow for the "self-assembly" of linkage structures by satisfying constraint equations using a critically damped function [7]. Their method is of order $O(n^3)$ computational complexity, where n is the number of constraints. This can be reduced to approximately $O(n^2)$ using a sparse matrix solution [8].

Witkin and Kass describe *spacetime constraints* in which the constraint equations are solved not only for the joint/link geometry, but also for the applied control forces [9]. Since the constraint equations are solved simultaneously for the entire time span of the simulation, energy (or some other function) can be minimized over all of time, though at a large computational cost. Like other constraint methods, forces that are not physically realizable may be produced to satisfy the constraints. Optimization techniques, such as spacetime constraints, contrast sharply with forward dynamic simulators, which require active force control to produce motion, thus encouraging the development of coordination strategies.

The non-constraint methods fall into two main categories: ones that form and solve a set of simultaneous equations for the accelerations, and ones that form a recursive relationship propagating force and movement information through the linkage, solving directly for accelerations. The first type of formulation typically yields $O(n^3)$ complexity, because solving the n simultaneous equations requires a matrix inversion. The recursive formulations, however, can reach $O(n)$ complexity, although they can be more difficult to develop, and the cost of the computations per link is higher than the simultaneous equation methods. Articulated figures with few joints are less expensive to compute using simultaneous equations; figures with many joints are more efficiently computed using the recursive methods. The Walker-Orin method is the most efficient of the simultaneous methods [10], and Featherstone's Articulated Body Method (ABM) is the most efficient of the recursive methods [3]. When $n \geq 9$, Featherstone's method is more efficient than the Walker-Orin algorithm.

Wilhelms describes a dynamics simulator based on the Gibbs-Appell formulation [11]. She and her co-workers note that while dynamic simulation can produce realistic motion, it does not simplify coordination and control, since it requires the specification of joint torques, rather than joint angles, without providing a means for determining the proper torques to apply. Because of this, they have developed an interactive, graphical editor, Virya, to allow the user to iteratively develop functions for useful force control.

Armstrong has developed a recursive dynamic formulation, which has approximately the same computational efficiency as the ABM [12]. Armstrong, *et al.*, have experimented with this method to simulate a moving human figure in near-real time [13]. However, the Armstrong method is only capable of efficiently simulating spherical joints, unlike the ABM which allows fully general joint types.

Another efficient recursive method has been formulated by Lathrop [14]. The advantage of Lathrop's method over Featherstone's is that it allows for kinematic constraints at the end-effectors. Lathrop's method can also be extended to handle kinematic loops [8].

Here we would like to distinguish between two simulation paradigms. Optimization and constraint techniques can be termed *key-event simulation*, because specific events must be satisfied. This is in contrast to *forward simulation* in which a system is established, and then simulated forward in time [15]. Controlling techniques for key-event simulation require global knowledge about the system, sometimes over the entire time span of the simulation. Forward simulation controllers, however, need only partial knowledge of the world, perhaps derived from a simulated sensor. The more sophisticated the controller, the more knowledge of the world it will require.

External influences, such as interactive input, can be easily incorporated into a forward simulation at any point in time, unlike methods such as spacetime constraints, which must be solved over the entire span of time.

In general, when it has been addressed at all, the coordination problem has been approached in two ways. Either interactive means are provided for the user to iteratively develop functions of time-varying forces, or mechanisms have been developed for defining and satisfying motion constraints. Interactive coordination methods are inadequate for all but the simplest motions, because complex, adaptive behavior is simply not amenable to "hands-on" control [16].

2.2. Human and Animal Motor Behavior

Natural gait has been the focus of much research for many years. Muybridge [17;18], Hildebrand [19], and others have cataloged and analyzed the stepping patterns of the legs during different mammalian gaits, such as galloping, cantering, trotting, etc. An analysis of insect and vertebrate locomotion indicates a hierarchical arrangement of simple control mechanisms which control stepping and stance [20] [21] [22] [23]. Our computational model of gait is based on these hypothesized biological mechanisms.

Bizzi, *et al.*, analyze the mechanical properties of the musculo-skeletal system and argue that motion is controlled by tuning the spring-like properties of muscles during movement [24]. Theoretical and clinical studies by Kugler, *et al.*, and by Robertson and Halverson, including studies of human locomotion, have examined the notion that coordinated motion arises more from the mechanical constraints on the physical system than from central motor programs [1;25]. We use the notion of tuned springs as a mechanism for controlling joint motion, as we describe in Section 3.3.

Beer, *et al.*, employ a heterogeneous neural network to simulate the stepping patterns exhibited by the cockroach [26;27]. Their work is intended primarily for the detailed study of the neural mechanisms of motor control, and provides only limited, 2D graphical output.

2.3. Walking Machines

Robotics research has resulted in fundamental algorithms for dynamic simulation and control, though in general, simulated articulated models in computer graphics have more kinematic DOFs than robot devices. McGhee developed an autonomous 6-legged robot vehicle which could employ a number of gaits to negotiate rough terrain with areas marked as forbidden [28]. Another hexapod vehicle was later developed by Sutherland; its gait control system is similar to the one presented here [29;30]. Legged robots designed by Raibert employ dynamic balance – i.e., they can run and hop without enough legs always on the ground to statically support the body [31]. McGhee and his co-workers have developed mathematical tools for analyzing multi-legged gait, and they use these tools for designing gait control algorithms for a very large six-legged vehicle [32].

2.4. Graphical Simulation of Gait

Computer animation and simulation of animal locomotion has involved primarily kinematic control. Zeltzer describes the use of finite state control to simulate adaptive human walking over planar and uneven terrain [33]. Girard reports the use of inverse-kinematics to interactively define gaits for legged animals [34].

Girard's gait sequencer is based on McGhee's analytical gait descriptions, so that gait changes are induced by explicitly varying the gait parameters – unlike our model, as we will show, in which smooth gait changes result naturally from variations in the overall gait frequency. Girard's model also incorporates some dynamic elements for added realism. However, the overall motion does not have a dynamic basis. Sims also employs inverse kinematics and dynamic elements, to simulate various adaptive gaits over uneven terrain [35]. Sims' gait sequencer is also based on McGhee's analytical gait description. Recently, Bruderlin has implemented a hybrid system in which limited dynamics and specialized knowledge about the kinematics of biped gait are applied to simulate human walking [36]. Our system, unlike any of these, makes use of a general-purpose rigid-body dynamics module, as well as a gait controller based on biological mechanisms.

3. A DYNAMIC LOCOMOTION SIMULATOR: *CORPUS*

Corpus is a system for simulating the forward dynamics of articulated figures, with gait control mechanisms, force-producing motor programs and rendering support¹. *Corpus* is implemented in C, and uses several support libraries, also implemented in C. There are three main procedural components (see Figure 1): a *dynamic simulator*, a *gait controller* and *motor programs*. In addition, a *structural description* specifies the mechanical and geometrical parameters of the hexapod. A scripting language is provided for defining jointed figures, and for controlling each of these components.

Dynamic simulation, using Featherstone's Articulated Body Method, forms the foundation of the system. The gait controller, based on the coupled oscillator model with reflexive feedback, coordinates the sequences of stepping and stance for the hexapod. The motor programs, based on exponential spring and damper combinations, generate the forces required for stepping and stance. The structural component contains descriptions of the kinematic structure of the links and joints, the mass and inertia of the links, the gait and motor program parameters, and finally, the graphic parameters necessary for rendering the component objects. The graphics system is not directly involved in the creation of locomotion, but is required for display and animation of the simulation.

3.1. The Dynamics Simulator

Our implementation of the Featherstone ABM algorithm is generalized and can compute the dynamics of any articulated figure with rigid links arranged in a branching structure, without internal closed loops (loops may be approximated using spring closure forces). Joints may be rotary, prismatic, or screw, and multiple-DOF joints, combining these types, may be represented.

We chose the Featherstone algorithm because it is an accurate and efficient recursive formulation for forward simulation, and because it can accommodate a variety of joint types. The algorithm is formulated in *spatial notation* – developed by Featherstone and based on screw calculus [37] – which essentially unites the rotational and translational aspects of motion into a single vector quantity. A complete treatment of spatial notation and the ABM algorithm is beyond the scope of this paper. The interested reader is directed to the work of

1. A "corpus" is the body of a man or animal (Webster's 7th), and thus the articulated-figure simulator, *corpus* is so-named.

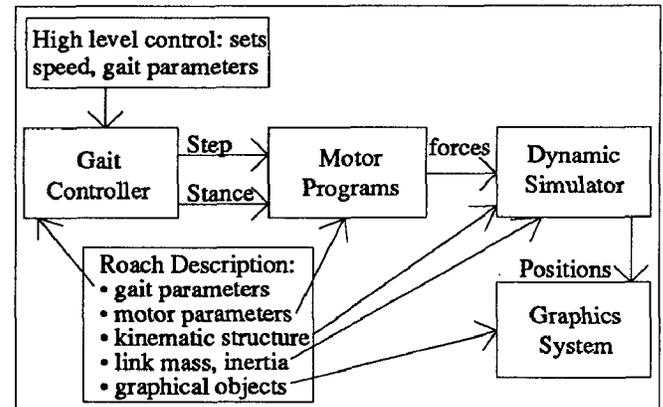


Figure 1: Block diagram showing the procedural and structural components of the *corpus* system.

Featherstone, and for discussions of spatial algebra, the ABM algorithm, as well as a more detailed discussion of implementation and numerical issues, the reader is referred to McKenna [38].

The key feature of the Articulated Body Method is the concept of the *articulated body inertia*. Just as the inertia tensor of a rigid body determines its acceleration when the applied force is known, the articulated body inertia tensor of a rigid body, within an articulated figure, determines the acceleration when the applied force is known. Formally, the equation of motion for an unconstrained rigid body is:

$$\hat{f} = \hat{I} \hat{a} + \hat{p}^v \quad \text{Eq. 1}$$

where \hat{f} is the applied force, \hat{I} is the inertia tensor, \hat{a} is the acceleration and \hat{p}^v is the bias force (centripetal and Coriolis) produced by the body's velocity. The "hat" ($\hat{}$) denotes quantities expressed in spatial notation. The equation of motion for a rigid body in an articulated figure reads:

$$\hat{f} = \hat{I}^A \hat{a} + \hat{p} \quad \text{Eq. 2}$$

where \hat{I}^A is the articulated body inertia tensor, and \hat{p} is the bias force which incorporates the \hat{p}^v from above, as well as joint reaction forces. The articulated body inertia \hat{I}^A gives directional properties to the *apparent* mass of a body.

Briefly, the algorithm progresses as follows: first the algorithm passes from the leaf bodies of the figure tree inward to the root body, accumulating the articulated body inertias \hat{I}^A and bias forces \hat{p} . Then Eq. 2 is used to compute the acceleration of the root body as in:

$$\hat{a} = (\hat{I}^A)^{-1} (\hat{f} - \hat{p}) \quad \text{Eq. 3}$$

Finally, the algorithm passes from the root out to the leaves computing the accelerations of the joints.

External forces and joint forces are specified before the dynamics algorithm begins; force generators include gravity, collisions, motor program springs, joint dampers, and spring connections to other fixed and moving bodies.



Once the accelerations have been computed by the dynamics algorithms, they must be numerically integrated to compute velocities and positions; *corpus* uses fifth order, adaptive Runge-Kutta [39]. Because the integrator must be able to back up in time in order to adapt to less stable conditions, the integrator structure was modified to support the storage of certain time-varying state information which must be passed to the dynamics algorithm.

The dynamics code in *corpus* is a straightforward implementation of the ABM, computed in body-local rather than world coordinates. This allows for some optimizations, provides more intuitive values for certain spatial quantities, and improves the accuracy of the integration.

Collision and contact in *corpus* is handled through spring forces. When a collision is detected between two bodies, forces are applied to each body as a function of penetration depth in the direction of the collision normal. Linear or exponential spring functions can be used to create the forces; exponential springs are typically employed because deep penetration is strongly resisted by the exponential rise in force. Damping, a force proportional to the penetration velocity in the direction to oppose that velocity, can be specified to create energy loss during collisions. However, energy loss is more directly modeled in *corpus* using the *coefficient of restitution*, ϵ , as in [40]. The coefficient describes the elastic properties of the collision and normally ranges from 0 (complete energy loss) to 1 (completely elastic collision). The calculated collision force is scaled by the coefficient when the colliding bodies are moving away from each other, yet are still interpenetrating. Friction is modeled in *corpus* by applying a force equal to the collision normal force, scaled by the coefficient of friction, in the direction opposed to the tangential sliding motion.

Collision detection is handled in several ways in *corpus*. The most basic method detects interpenetration of body bounding boxes. The simple geometry allows for rapid execution when only rectangular solids are employed (as with level terrain and the roach model). Another rapid method detects bounding box penetration against a height field, suitable for detection of collisions of the roach model with uneven terrains. Finally, a general algorithm is available for detection of interpenetration of arbitrarily shaped objects.

3.2. The Gait Controller

In order to move from place to place, an animal must coordinate its limbs to bring about coherent motion. Legs are alternately controlled between *step* and *stance*. Stepping brings the leg up and forward, while stance supports the body and drives it forward. The overall sequence of the various legs stepping and standing is termed the *gait*.

Wilson analyzed the stepping patterns cockroaches exhibited under a variety of conditions, and proposed five rules which describe the gait behavior of many insects [41]:

- 1) A wave of steps runs from rear to head (and no leg steps until the one behind is placed in a supporting position).
- 2) Adjacent legs across the body alternate in phase.
- 3) Stepping time is constant.
- 4) The frequency with which each leg steps varies.
- 5) The interval between steps of adjacent legs on the same side of the body is constant, and the interval between the stepping of the foreleg and hindleg varies inversely with the stepping frequency.

Wilson made hypotheses about the neurological mechanisms which could generate these rules, and his ideas were confirmed by the experimental work of Pearson [23]. Each leg in the cockroach has a pacemaker or *oscillator*, which rhythmically triggers the leg to step. The oscillators are coupled together, and their interaction generates the various gaits.

In addition to the coupled oscillators, *reflexes* also play an important role in gait generation. Reflexes can both trigger or retard the stepping of limbs. In nature, the cockroach *step reflex* causes a leg to step when hair receptors detect that the leg has nearly reached its maximum rearward extension. Another cockroach reflex employs cuticle stress-receptors, which measure the load that a leg is bearing, and prevents a leg from stepping if it is supporting the insect. In general, reflexes reinforce the stepping pattern generated by the coupled oscillators, while increasing the adaptability of the creature under changing environmental conditions. Reflexes seem to play an even more important role during locomotion over uneven terrain. A study by Pearson of locusts walking over uneven terrain shows that a fixed stepping pattern is not employed over rough terrain [42]. To find suitable footholds, the legs employ searching tactics, and an *elevator reflex* causes the leg to lift higher if it encounters an obstacle during a stepping movement.

In *corpus*, each leg is assigned an oscillator, which periodically triggers stepping activity (see Figure 2a). The coupling between oscillators is modeled as phase and time relationships which the oscillators maintain among each other. These relationships are mathematical translations of the stepping rules observed in the cockroach by Wilson [41]. The oscillators are constrained to match a master frequency, such that the coupling rules generate differing gaits and walking speeds as the master oscillator frequency is varied. At slow oscillator frequencies, slow *wave* gaits are generated. As the oscillator frequencies increase, faster wave gaits result until finally the *tripod* gait is generated. As the oscillator frequencies smoothly change, a smooth gait change is effected.

Reflexes are modeled as conditional units (see Figure 2b). When a certain condition is met, the reflex can inhibit or trigger different actions. For example, the step reflex triggers stepping when a leg is extended beyond a specified angle, which prevents over-extension of the legs. A load bearing reflex inhibits stepping if a leg is currently bearing too much weight. This prevents the hexapod from lifting a leg while it is supporting the body.

The oscillators and reflexes trigger the stepping motor programs for the legs. Once stepping is initiated, it continues to completion and stance begins again. The gait controller only generates the pattern of stepping, and is not directly responsible for the movements of the legs or body. However, the movements of the legs, due to the motor programs and dynamic simulation, provide feedback into the gait controller through the reflexes.

3.3. Motor Programs

The dynamic motor programs are responsible for delivering forces, through the joints of the hexapod, to create the movements required for locomotion. There are two motor programs: step and stance. The stepping program must compute the forces necessary to lift the leg up and forward, and place it in a position to take up the load of the body when stance begins. The stance program supplies the forces needed

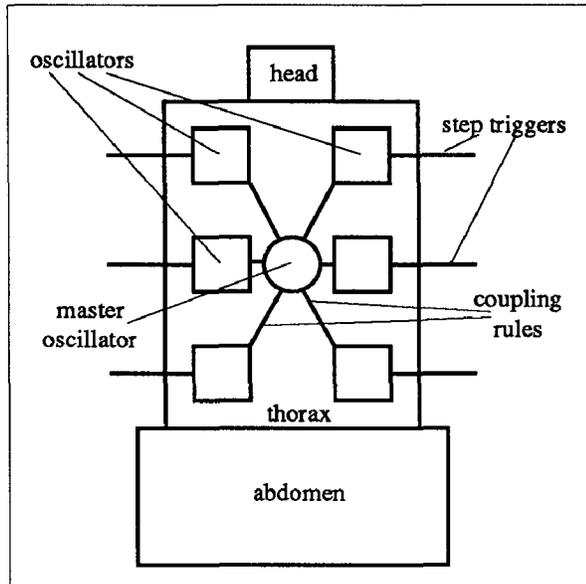


Figure 2a: The coupled oscillator configuration

to support the body via the legs, and propel it forward. Stepping programs are triggered by the gait controller, as described above. Stance programs automatically begin when stepping has completed.

In biological systems, the basic producer of bio-mechanical forces is the muscle. McMahon [43] contains an excellent review of the force-producing properties of muscle, under varying types of stimulation and external influences. Starting with the assumption that muscles are tunable, spring-like force generators, motor control researchers have come up with an *equilibrium-point hypothesis* to explain how controlled movements are produced [24; 44]. This model treats the muscle, along with its feedback system, as a single, tunable unit, with measurable, spring-like properties. Postures are controlled by establishing an equilibrium between agonist and antagonist muscle groups. This equilibrium configuration forms a point (in a controlling space) which can be specified by the neuromuscular system. The equilibrium-point hypothesis states that movements are produced by changing the equilibrium point from one posture to another. Hogan describes a *virtual trajectory* of equilibrium points which control movements [45].

The dynamic motor programs in *corpus* create forces by using exponential springs. As their name implies, these springs have an exponential relationship between the displacement, x , of the spring from its rest position (or angle) and the force generated, f , such that:

$$f = \alpha (e^{\beta |x|} - 1)$$

where α controls linear strength, and β controls exponential rise. The exponential response creates a steep potential well; with a large displacement, the force becomes extremely high. The β parameter controls the width of the well, and the α parameter controls how fast a well of a given width will linearly rise. When an exponential spring is used for position control, the DOF it controls will very likely stay within the lower parts of the well, since the forces grow so large outside of the lower region.

A motor program controls the rest position (or angle) of an exponential spring over time, which causes the force potential-well to travel along the DOF, "dragging" the controlled limb along with it. The rest position is modified using a linear

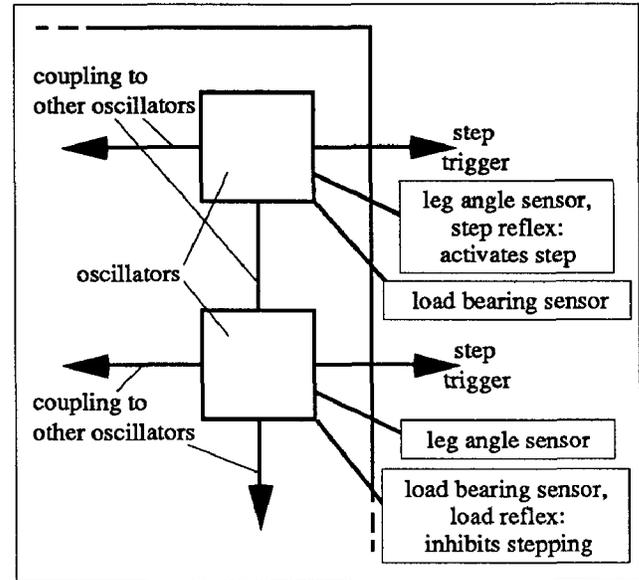


Figure 2b: reflex feedback to the oscillators

interpolation from the current position to the target position. In more physical terms, the rest position travels with a constant velocity to the target. This is basically open-loop control, which is appropriate for so-called *patterned* gaits, but inappropriate for *free* gaits – in which safe footholds must be found – or other movements which require positional accuracy, such as reaching and grasping.

The potential wells created by the springs lead to a *compliant* system, which allows the final motion to fall within a range of possible motions. For example, to negotiate uneven terrain, a kinematic system would need to compute the leg *joint angles* required to place the feet on the varying heights of the terrain surface. Using a dynamic, compliant system however, the legs of our simulated insect can automatically conform to the terrain (see Figures 6 and 8).

A disadvantage of using exponential springs is that they can create a stiff system. As the force response of the springs is pushed further and further up the steep walls of the potential well, the numerical sampling of the integrator must take smaller and smaller time steps to get an accurate result. Linear springs would not create such a stiff system for a given force output, but exponential springs have the advantage that at small displacements, they are less stiff than linear springs. In addition, linear springs need to be very strong to create similar forces to the exponential springs at large displacements.

3.4. Structural Description

The kinematic structure of the hexapod was derived from insect physiology references [46]. Diagrams of the insect *Blatta* were used to parametrize the sizes of the limb parts of the hexapod[47]. A reproduction of one of the diagrams, along with a view of the resulting articulated, solid model is shown in Figure 3. The lengths and widths of the limb parts were measured, and a rectangular solid was constructed to represent each part. No further refinement in the shape of the limb and body parts beyond the rectangular solid was attempted; our study focuses instead on the basic motions and physical parameters involved in locomotion.

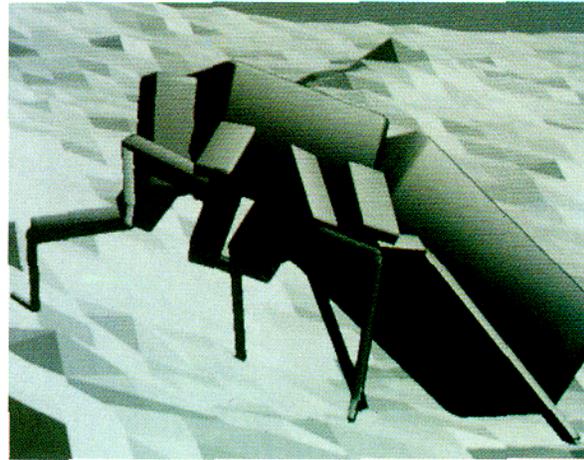
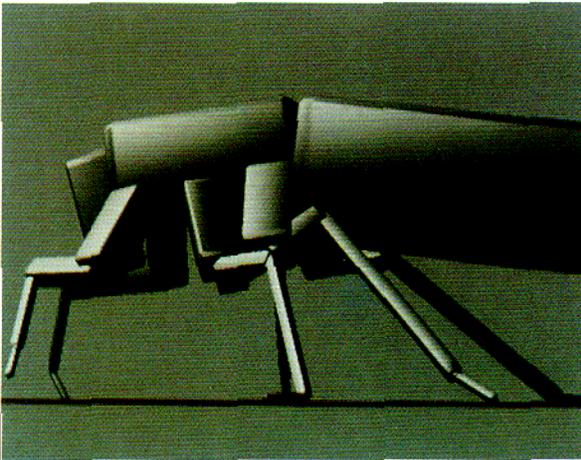
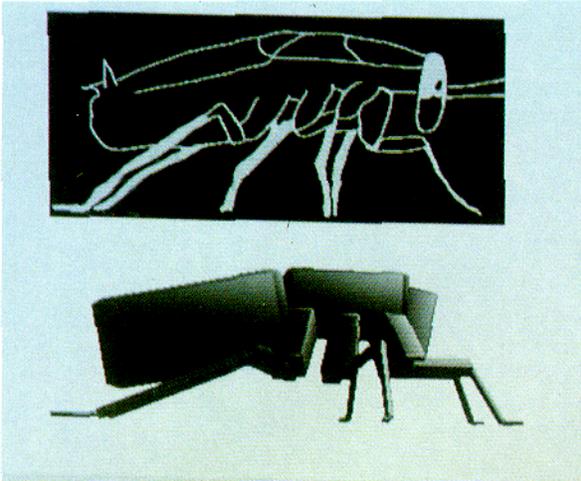


Figure 3: (upper left) Diagrams of the insect *Blatta* were used to parametrize the hexapod model. An example diagram is shown with the resulting figure.

Figure 6: (upper right) The hexapod reclines within the curved surface of its "Pod". (still from the animation *Grinning Evil Death*).

Figure 7: (middle left) The hexapod is shown employing the tripod gait over level terrain.

Figure 8: (middle right) The hexapod is depicted using the wave gait over uneven terrain. Mechanical compliance in the limb joints allows the hexapod to adapt to the different heights of the terrain.

Figure 9: (lower left) A still from the animation *Grinning Evil Death* shows the hexapod interacting with other dynamic elements. The wires are dynamic linkages, which are allowed to break when loop closure forces exceed a given limit. Collision forces are generated at points of contact between the roach and the wires.

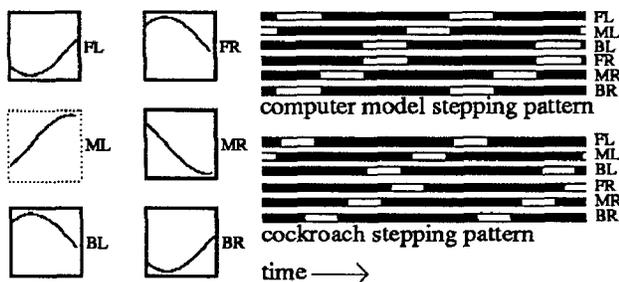


Figure 4: The coupled oscillators produce a wave gait at low oscillator frequencies. The activity of the oscillator model is shown to the left. When an oscillator reaches its peak, its leg is triggered to step, indicated by a dotted box. The stepping pattern of both the computer model and the biological cockroach are shown to the right. White indicates step, and black stance. The cockroach stepping pattern, adapted from Pearson [23], depicts a slightly faster walking speed than the computer model stepping pattern.

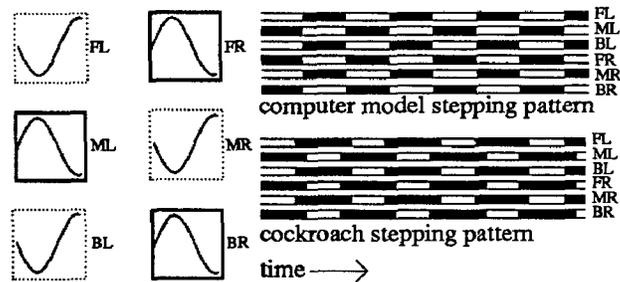


Figure 5: The coupled oscillators produce the tripod gait at the maximum oscillator frequency. The cockroach and computer model stepping patterns are essentially identical, except that the cockroach has a longer stance time than step time. In the computer model, the step and stance time were set to be identical during the tripod gait in order to drive a standing leg backwards at the highest velocity. However, stability might be increased by allowing the standing leg to take up more of the body weight before its neighbors step, which would result in a stepping pattern more like the biological cockroach's. (cockroach stepping pattern adapted from Pearson [23].)

The overall scaling of the roach gives it a total length of approximately 2.9 cm. The density of the body and limb parts was set to the density of water, 1 gm/cm^3 , since animal tissues in general are composed mostly of water. The total mass of the hexapod is 2.1 gm.

A set of control parameters determine the basic gait features and the motor program parameters. Constant gait features are the stepping speed, the time between stepping of adjacent legs on the same side of the body, the number of legs and default values for other parameters such as oscillator frequency. The motor programs for step and stance define what joint angles are traversed by the exponential springs during those actions. These programs are *tuned* via a trial-and-error method to determine appropriate spring strengths and joint angle values. In general, this trial-and-error approach is not the appropriate method to determine the operating dynamic parameters, since it requires an "expert" tuner to make "educated" guesses as to the parameters, based on the experience gained from previous experiments. In some sense, the expert tuner acts as natural selection in an "evolutionary" process which increases the robustness of the locomotion. An alternative to the manual tuning process would be to employ automatic calibration. The operating parameters for the motor programs could be determined by an inverse dynamics or constrained optimization technique in a calibration phase before the primary simulation. Alternately, an automatic evolutionary process could be employed in which successive, random changes are made to the motor program parameters (as well as other structural descriptions). The resulting simulations would be evaluated using measurement criteria, such as walking speed, distance covered, and energy expended.

4. RESULTS AND ANALYSIS

4.1. The Gait Controller

The computational model of the coupled oscillators produce stepping patterns which appear very similar to the recorded patterns of insect stepping [41;23]. For slow oscillator frequencies, the wave gait results (see Figure 4). The fastest allowed oscillator frequency produces the tripod gait (see Figure 5). Smooth changes in oscillator frequency result in smooth changes in gait.

The step reflex and load bearing reflexes function correctly, but require calibration. They should not function during undisturbed walking, but should instead reinforce stable stepping patterns under disturbances. The calibration procedure is to observe and analyze undisturbed walking, and then set reflex trigger values beyond the norm. Different leg pairs (front, middle, and back) will require different values, since their ranges of motion are different, and they support different loads. This calibration has not yet been performed for the hexapod model. However, these two reflexes have been studied for a simple kinematic hexapod [48]. The step reflex increases the robustness of the gait, especially during turning and speed changes. The load-bearing reflex (implemented in the kinematic model as a table lookup of stable stepping patterns) increases stability when limbs are missing, and prevents the step reflex from triggering a supporting leg to step. To date, we have studied adaptive locomotion not through active control, but rather through the mechanical compliance of the physical simulation.

4.2. Walking Experiments

Using the initial joint and spring angles established from the *Blatta* diagrams, the hexapod was "dropped" onto level ground in several dynamic simulations. The first attempts employed linear springs at the joints, and on every attempt the hexapod would collapse, as the supporting forces generated at the joints were not strong enough. Increasing the spring constants only resulted in a very stiff system, without providing enough support for the figure to stand. When exponential springs were introduced at the joints in place of the linear springs, they created forces sufficient to support the hexapod as it was dropped on the ground. In addition, while the hexapod was falling through the air, the integrator only slightly subdivided the frame rate – far less than with the strong linear springs – since the exponential springs generate less force at low displacements.

During the first walking experiment, the initial posture was found to be too low, and the hexapod dragged its abdomen along the ground behind it. Although in nature the cockroach frequently drags its abdomen along the ground [47], we desired

a model of locomotion in which the body was fully supported as in many other insects. Therefore the posture was raised by using joint motor programs to move the exponential spring rest angles to values which further extended the limbs. These spring angles were used as the new initial configuration for further walking experiments.

Dozens of walking simulations have been executed, often successively "tuning" the action of the motor programs or other parameters. For example, the step program originally did not lift the foot fast enough or high enough to avoid dragging it along the ground for much of the stepping time, so the motor program was modified to lift the leg up higher, and more rapidly at the beginning of the step. Typical motor program parameters for the hexapod are available in [38].

Figure 7 shows the hexapod employing the tripod gait over level terrain. The interval between steps of successive legs employed was 50 msec, compared to approximately 120 msec for the beetle *Chrysomela* which has a "relatively long" stepping interval [47]. The walking speed exhibited by the hexapod was approximately 5.5 cm/sec. Insects show a wide variety of walking speed, varying from 2.0-9.8 cm/sec in the Earwig, 3.2-17.5 cm/sec for *Blatta*, and 1.0-20.0 cm/sec for the cockroach, *Periplaneta*. The walking speed of our simulated roach falls well within these ranges, but is considerably slower than real insects walking at their top speeds. This experiment employed a sliding model of friction with a fairly low coefficient of friction (0.7).

A different walking experiment, also employing the tripod gait, used a ground contact model in which the "feet" were modeled as having sticky pads, under active control of the hexapod, as in the honey-bee and many other insects [46]. During stance, the feet would stick to the ground using exponential springs. The springs were allowed to break, if the force rose above a specified limit, allowing the feet to slide slightly and stick again. The walking speed of the hexapod increased to approximately 8.0 cm/sec, using the sticky foot model. Constraint-based methods, especially Lathrop's, would be appropriate to simulate these constrained kinematic foot placements.

An interesting observation is that our hexapod exhibits a side-to-side "wobble" as it progresses forward, using the tripod gait. In fact, the same sort of zig-zag path is seen in real insects [46]. The phenomenon can be explained when the propulsive forces are analyzed. The front supporting leg acts as a tractor, pulling the center of mass forward, and towards the point of support. The rear supporting leg (on the same side of the body as the front support) pushes the body forward, and produces either clockwise or counterclockwise turning forces, depending on whether the line of force produced by the limb passes in front of or behind the body's center of mass. At the beginning of stance, the rear leg will tend to rotate the body in the same direction as the front leg. As stance continues and the rear foot moves back relative to the body, the line of force produced by the leg will shift further and further forward, and its turning forces will tend toward the opposite direction. The middle supporting leg, on the opposite side of the body, serves to support that side, propel the body forward, and to counteract part of the rotary forces produced by the other two supporting limbs.

Locomotion over uneven terrain is shown in Figure 8. The "sticky-foot" model of contact was used for this simulation, to prevent the hexapod from sliding down the hill. The hexapod

adapts to the terrain purely by the mechanical compliance provided by the springs and dampers in the legs. The stepping and stance motor programs were not modified for the terrain; a more complete system should adapt its motor control for different environmental conditions. However, it is interesting to note how dynamic simulation and mechanical compliance can lead to adaptive behavior, without special planning.

The computation time involved in simulating the walking motion of the hexapod is relatively high, especially compared to kinematic models. On a Hewlett-Packard Series 9000 Model 835 (a RISC based workstation, rated at 12 MIPS) one video frame at 1/40 real time (1/1200 sec simulation time) takes approximately 4 minutes of computation time. The dynamics algorithm is called approximately 600 times in that interval by the adaptive step-size integrator. A simple kinematic model of the hexapod operates in real time, but has fewer degrees of freedom (20 DOF vs. 38 DOF) and does not display complex, realistic motion. The dynamics code does not currently take advantage of several numerical optimizations, which could increase speed by an order of magnitude. In addition, a stiff-system integrator could increase speed greatly by saving many calls to the dynamics algorithm.

5. FUTURE WORK

The number of legs can simply become a parameter to the gait controller. We have used the same coupled oscillator paradigm to generate realistic biped and quadruped gaits, though only in kinematic simulations. Insects can employ wave and tripod gaits which always provide at least three support points at all times during the gait cycle, i.e., they rely on *static balance*. Biped and quadrupeds, however, use mostly *dynamic balance*, in which the figure is falling from support point to support point. We intend to study the interaction of the mechanics of the figure with the coordination strategy in order to develop a dynamic biped locomotion system.

6. SUMMARY

The realistic simulation and animation of the motions of human and animal figures has long been a goal of researchers in computer graphics. We have presented a dynamic locomotion simulator in which the coordination of a kinematically complex virtual insect is automatically generated by a gait controller, and realistic motions of the limbs are produced by stepping and stance motor programs which apply appropriate forces to the limbs. Motion is accurately and efficiently computed by our implementation of the Featherstone Articulated Body Method. The simulation agrees well with the observed behavior of insects. The coupled oscillator model of gait coordination is general, and can be used to control biped and quadruped gaits.

ACKNOWLEDGEMENTS

The *corpus* system makes use of several large support libraries coded in C at the Computer Graphics and Animation Group, including *rendermatic*, a rendering library with geometric collision detection, by Brian Croll and David Chen; *retepmatic*, a rendering package with additional functionality by Peter Schröder, and *robotlib* an inverse kinematics and matrix manipulation package, by David Chen.

In addition, Bob Sabiston must be given ample credit for co-designing the stills from *Grinning Evil Death*.

BIBLIOGRAPHY

- [1] Kugler, P. N., J. A. S. Kelso and M. T. Turvey. On the Concept of Coordinative Structures as Dissipative Structures: I. Theoretical Line. *Tutorials in Motor Behavior*. Amsterdam, North-Holland (1980).
- [2] Featherstone, R. The Calculation of Robot Dynamics Using Articulated-Body Inertias. *Robotics Research* 2,1 (1983), 13-29.
- [3] Featherstone, R. *Robot Dynamics Algorithms*. Kluwer Academic Publishers (1987).
- [4] Schröder, P. *The Virtual Erector Set*, Master's Thesis, Massachusetts Institute of Technology (1990).
- [5] Barzel, R. and A. H. Barr. Controlling Rigid Bodies with Dynamic Constraints. ACM SIGGRAPH '88 Course Notes #27: Developments in Physically-Based Modeling, Section E (1988).
- [6] Isaacs, P. M. and M. F. Cohen. Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions and Inverse Dynamics. *Computer Graphics* 21,4 (July 1987), 215-224.
- [7] Barzel, R. and A. H. Barr. A Modeling System Based on Dynamic Constraints. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1988) In *Computer Graphics* 22,4 (August 1988), 179-188.
- [8] Schröder, P. and D. Zeltzer. The Virtual Erector Set: Dynamic Simulation with Linear Recursive Constraint Propagation. Proceedings of the 1990 Symposium on Interactive 3D Graphics (Snowbird, Utah, March 1990). In *Computer Graphics* 24, 2 (1990), 23-31.
- [9] Witkin, A. and M. Kass. Spacetime Constraints. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1988) In *Computer Graphics* 22,4 (August 1988), 159-168.
- [10] Walker, M. W. and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. Proceedings of Joint Automatic Contr. Conf. (Charlottesville, VA, 1981).
- [11] Wilhelms, J. Using Dynamic Analysis for Realistic Animation of Articulated Bodies. *IEEE Computer Graphics and Applications* 7,6 (June 1987), 12-27.
- [12] Armstrong, W. W. Recursive solution to the equations of motion of an n-link manipulator. Proceedings of 5th World Congress Theory Mach. Mechanisms (Montreal, 1979) Volume 2, 1343-1346.
- [13] Armstrong, W. W., M. Green and R. Lake. Near-Real-Time Control of Human Figure Models. *IEEE Computer Graphics and Applications* 7,6 (June 1987), 52-61.
- [14] Lathrop, R. H. Constrained (Closed-Loop) Robot Simulation By Local Constraint Propagation. Proceedings of 1986 IEEE Int. Conf. on Robotics and Automation (San Francisco, 1986) Volume 2, 689-694.
- [15] Zeltzer, D., S. Pieper and D. Sturman. An Integrated Graphical Simulation Platform. Proceedings of Graphics Interface 89 (London, Ontario, 1989), 266-274.
- [16] Zeltzer, D. Towards an Integrated View of 3-D Computer Animation. *The Visual Computer* 1,4 (December 1985), 249-259.
- [17] Muybridge, E. *The Human Figure in Motion*. New York, Dover (1955).
- [18] Muybridge, E. *Animals in Motion*. New York, Dover (1957).
- [19] Hildebrand, M. Analysis of Tetrapod Gaits: General Considerations and Symmetrical Gaits. *Neural Control of Locomotion*. New York, Plenum Press (1976).
- [20] Gallistel, C. R. *The Organization of Action: A New Synthesis*. Hillsdale, New Jersey, Lawrence Erlbaum Associates (1980).
- [21] Gelfand, I. M., V. S. Gurfinkel, M. L. Tsetlin and M. L. Shik. *Models of the Structural-Functional Organization of Certain Biological Systems*. Cambridge, MIT Press (1971).
- [22] Grillner, S. Locomotion in Vertebrates: Central Mechanisms and Reflex Interaction. *Physiological Reviews* 55,2 (April 1975),
- [23] Pearson, K. The Control of Walking. *Scientific American* 235,6 (December 1976), 72-86.
- [24] Bizzi, E. Central and peripheral mechanisms in motor control. *Tutorials in Motor Behavior*. North-Holland Publishing Co. (1980).
- [25] Robertson, M. A. and L. E. Halverson. The Development of Locomotor Coordination: Longitudinal Change and Invariance. *Journal of Motor Behavior* 20,3 (1988), 197-241.
- [26] Beer, R. D., L. S. Sterling and H. J. Chiel. Periplaneta Computatrix: The Artificial Insect Project. Case Western Reserve University. Technical Report, TR 89-102. (January 1989).
- [27] Chiel, H. J. and R. D. Beer. A lesion study of a heterogeneous artificial neural network for hexapod locomotion. Case Western Reserve University. Technical Report TR-108. (February 1988).
- [28] McGhee, R. B. Robot Locomotion. *Neural Control of Locomotion*. New York, Plenum Press (1976).
- [29] Raibert, M. H. and I. E. Sutherland. Machines That Walk. *Scientific American* 248,1 (January 1983), 44-53.
- [30] Donner, M. D. *Control of Walking: Local control and real time systems*. PhD Thesis, Carnegie-Mellon University. (1984).
- [31] Raibert, M. H. *Legged Robots That Balance*. Cambridge, MA, MIT Press (1986).



- [32] Song, S. and K. J. Waldron. *Machines That Walk: The Adaptive Suspension Vehicle*. Cambridge, MA, MIT Press (1989). 1990) In *Computer Graphics* 24,2 (1990), 165-174.
- [33] Zeltzer, D. Motor Control Techniques for Figure Animation. *IEEE Computer Graphics and Applications* 2,9 (November 1982), 53-59.
- [34] Girard, M. and A. A. Maciejewski. Computational Modeling for the Computer Animation of Legged Figures. *Computer Graphics* 19,3 (July 1985), 263-270.
- [35] Sims, K. *Locomotion of Jointed Figures over Complex Terrain*, M.S.V.S Thesis, Massachusetts Institute of Technology. (June 1987).
- [36] Bruderlin, A. and T. W. Calvert. Goal-Directed, Dynamic Animation of Human Walking. Proceedings of SIGGRAPH '89 (Boston, Massachusetts, July 1989) In *Computer Graphics* 23,3 (July 1989), 233-242.
- [37] Ball, R. S. *A treatise on the theory of screws*. London, Cambridge Univ. Press (1900).
- [38] McKenna, M. A. *A Dynamic Model of Locomotion for Computer Animation*. Master's Thesis, Massachusetts Institute of Technology. (1990).
- [39] Forsythe, G. E., M. A. Malcolm and C. B. Moler. *Computer Methods for Mathematical Computations*. New Jersey, Prentice-Hall, Inc. (1977).
- [40] Moore, M. and J. Wilhelms. Collision Detection and Response for Computer Animation. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1988) In *Computer Graphics* 22,4 (August 1988), 289-288.
- [41] Wilson, D. M. Insect Walking. *Annual Review of Entomology* 11 (1966), 162-169.
- [42] Pearson, K. G. and R. Franklin. Characteristics of Leg Movements and Patterns of Coordination in Locusts Walking on Rough Terrain. *The International Journal of Robotics Research* 3,2 (1984), 101-112.
- [43] McMahon, T. A. *Muscles, Reflexes, and Locomotion*. Princeton University Press (1984).
- [44] Bizzi, E., W. Chapple and N. Hogan. Mechanical Properties of Muscle: Implications for Motor Control. *Trends in Neuroscience* (November 1982).
- [45] Hogan, N. The Mechanics of Multi-Joint Posture and Movement Control. *Biological Cybernetics* 52 (1985), 315-331.
- [46] Wigglesworth, V. B. *The Principles of Insect Physiology*. London, Chapman and Hall .
- [47] Hughes, G. M. and P. J. Mill. Locomotion: Terrestrial. *The Physiology of Insecta*. New York and London, Academic Press (1974).
- [48] McKenna, M. , S. Pieper and D. Zeltzer. Control of Virtual Actor: The Roach. Proceedings of 1990 Symposium on Interactive 3D Graphics (Snowbird, Utah,